**Set Up Multiple Slots in a Lex Chatbot:**

Welcome back to my AI x AWS series!

In this five-project journey, I’ve been building \*\*BankerBot\*\*, a chatbot that helps my imaginary bank’s customers check their account balance and transfer money between accounts.

So far, I’ve learned how to:

💬 Define intents to understand user requests

🔀 Provide response variations for a more natural conversation

🌟 Set up a custom slot type to capture user-specific details

🤝 Connect my chatbot with AWS Lambda for backend processing

🚁 Enable context carryover between intents for a seamless experience

Now, I’m ready to take the next step! What’s next for BankerBot? 🚀

In this final part, I'll create an intent that lets you transfer money between accounts and learn about a handy service called AWS CloudFormation.

**Step 1: Set up a new Lex chatbot**

In this, I’ll create a new BankerBot by just repeating Part 1 and Part 2 projects.

**Step 2: Create the new TransferFunds intent**

My users can check their account balance now, but what about transferring money between accounts?

This part is exciting because, for the first time, I need to handle two account types within the same intent! That means I have to make sure BankerBot correctly understands:

💰 The account to transfer from

🏦 The account to transfer to

💲 The amount to transfer

Setting this up is going to be fun-time to refine my slot types and logic to make sure everything runs smoothly!

In this step:

* Set up a new intent that uses two accountType slots.
* Create a new empty intent with the following properties:
  + Name: TransferFunds
  + Description: Help user transfer funds between bank accounts

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Sample utterances:

Can I make a transfer?

I want to transfer funds

I'd like to transfer {transferAmount} from {sourceAccountType} to {targetAccountType}

Can I transfer {transferAmount} to my {targetAccountType}

Would you be able to help me with a transfer?

Need to make a transfer

A screenshot of a computer

AI-generated content may be incorrect.

* Slots: add a new slot called sourceAccountType, with the prompt Which account would you like to transfer from? and the slot type **accountType**.

A screenshot of a computer screen

AI-generated content may be incorrect.

* Slot: add another new slot called targetAccountType, with the prompt Which account are you transferring to? and the slot type **accountType**.

A screenshot of a computer screen

AI-generated content may be incorrect.

* Slot: add another new slot called transferAmount, with the prompt How much money would you like to transfer? and the slot type **AMAZON.Number**.

A screenshot of a computer

AI-generated content may be incorrect.

Two of my slots have the same slot type!

It's totally possible for multiple slots to have the same slot type.

When two slots have the same slot type, it becomes important that you're using clear slot names, like sourceAccountType and targetAccountType, to make it easy to identify their differences.

A screenshot of a computer

AI-generated content may be incorrect.

* Now, I want to add confirmation prompts.

**What are confirmation prompts?**

Confirmation prompts typically repeat back information for the user to confirm. e.g. "Are you sure you want to do x?" If the user confirms the intent, the bot fulfills the intent

If the user declines, then the bot responds with a decline response that you set up.

* Scroll to the **Confirmation** panel.
* In the **Confirmation prompt** panel, enter the following:
  + Got it. So we are transferring {transferAmount} from {sourceAccountType} to {targetAccountType}. Can I go ahead with the transfer?
* In **Decline** response, enter:
  + The transfer has been cancelled.

A screenshot of a computer

AI-generated content may be incorrect.

* Scroll to the **Closing response** pane, and add this to the Message field:
  + The transfer is complete. {transferAmount} should now be available in your {targetAccountType} account.
* Now I’m ready for testing! Choose **Save intent.**
* Choose **Build**

A screenshot of a computer

AI-generated content may be incorrect.

* Choose **Test**.
* Now let's test out the chatbot! Ask the chatbot: I'd like to transfer money.

A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Delete resources:**

Time to clean up those resources - let's make sure we don't rack up any charges!

Steps:

* + 1. Delete the BankerBot
    2. Delete the Lambda function
    3. Delete Lambda function’s log files

Delete the BankerBot

1. Head to Amazon Lex console
2. Choose Bots on the left-hand sidebar
3. Choose the circle radio button next to BankerBot.
4. Choose Delete from the Action drop-down.
5. Choose Delete.

Delete the Lambda function

1. Head to the **AWS Lambda** console.
2. Choose **Functions** on the left-hand sidebar.
3. Choose the circle radio button next to **BankingBotEnglish**.
4. Choose **Delete** from the **Action** drop-down.
5. Choose **Delete**.

Delete your Lambda function log files

**What are Lambda function log files?**  
These log files are records of events that happen when AWS Lambda functions are running. These logs include details like errors, warnings, and informational messages that help developers understand how their functions are performing.  
In this project, log files have been produced each time your Lambda function was triggered i.e., when CheckBalance gives you a bank balance number.  
The logs are stored in CloudWatch, an AWS service used for monitoring and managing all the log data across all your services.

* Head to **CloudWatch** in the AWS console.
* Choose **Logs** on the left-hand sidebar.
* Choose **Log groups.**
* Select the checkbox next to **BankingBot**.
* Choose **Delete log group(s)** in the **Actions** menu.
* Choose **Delete**.

A computer screen shot of a computer

AI-generated content may be incorrect.

**Deploy your bot in seconds**

Oh wait, another step? I just **deleted my bot!**

No worries—I'm actually supposed to start this step from ground zero!

But now that I’ve recreated **BankerBot** multiple times using the **Amazon Lex console**, I can’t help but wonder…

🤔 **Is there a faster way to set things up?**  
🤯 **Do I really have to configure every intent manually each time?**

As engineers, we’re always looking for ways to **optimize and automate**. Instead of repeatedly setting up my bot through the console, I need a **better, faster, and more scalable approach**.

🥁 Introducing... **AWS CloudFormation!**

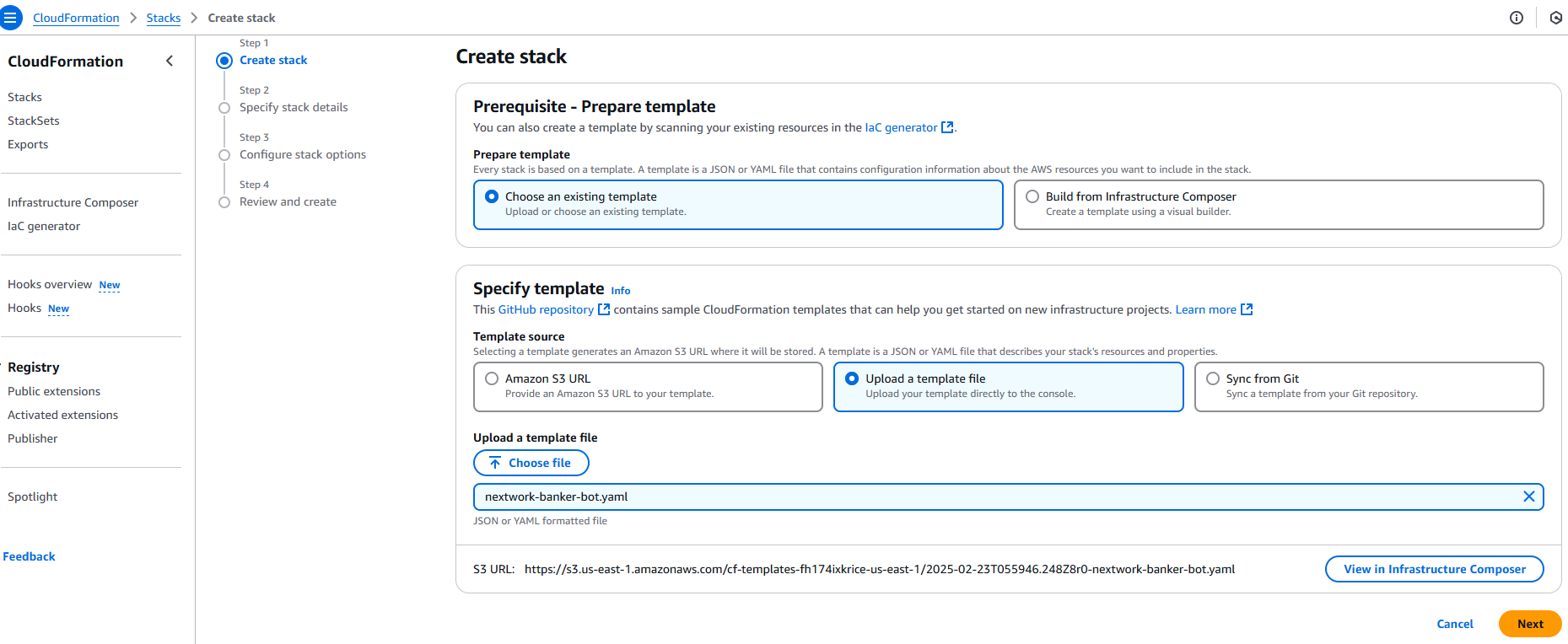
**What is AWS CloudFormation?**  
AWS CloudFormation is a service that gives you an easy way to create and set up AWS resources.

It's an **infrastructure as code** service - meaning you will use a file that describes all the resources you want to create and their dependencies as code. Then, you can use that template to create, update, and delete the entire stack of resources you described, instead of managing your resources individually

CloudFormation is a super time saver. Let's give it a try with our BankerBot!

**In this step, get ready to...**

* Use CloudFormation to deploy BankerBot in seconds.
* Resolve connection issues with your deployed bot.
* Here's the CloudFormation template you'll be using today!  
  👉 Download this: [nextwork-banker-bot.yaml](https://storage.googleapis.com/nextwork_course_resources/courses/aws/AWS%20Project%20People%20projects/Project%3A%20Create%20a%20chatbot%20with%20Amazon%20Lex%20/nextwork-banker-bot.yaml)
* Open up the file on your laptop and give it a read - can you recognize its content?
* Here's an excerpt that shows you how the **CheckBalance** and **FollowupCheckBalance** intents would be described in a CloudFormation template!
* Next, let's deploy the CloudFormation template and see it at work!
* Head to CloudFormation in your AWS Console.
* Choose **Create stack**.
* Choose **Choose an existing template**.
* Choose **Upload a template file**.
* Upload your CloudFormation template!



* Choose **Next**.
* For **Stack name**, you can enter **nextwork-banker-bot-name**
* Your stack name must be unique across your entire Region! Make your stack name unique if it's already taken.
* Choose **Next**.
* Choose **Next** again! We can skip Stack configuration in this simple implementation.
* Now you're on the final **Review** page. Scroll to the bottom and select the checkboxes under **Capabilities and transforms**.
* Choose **Submit**.
* Your stack might take around 5 minutes to create. Wait until the status of your **Stacks** page says **CREATE\_COMPLETE**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Head to your Amazon Lex console, and you'll notice that your bot is all created and ready for testing already.
* Click into **banker-bot-BankerBot**.
* Select **Intents** from your left-hand navigation menu.

A screenshot of a computer

AI-generated content may be incorrect.

* Connect your **TestBot Alias** with a Lambda function. Select the one with **V2** in the name.
* Select **Test**.
* Let's chat with our new chatbot! Use **Version 1** of your chatbot.
* You'll notice that it mostly functions just like our previous chatbot, but hmmm.. do you get this error too when trying to activate the **CheckBalance** intent? The chatbot would simply return **Intent CheckBalance is fulfilled**.

Does **CheckBalance** work after doing that?

If it doesn’t and you get hit with a 🚨 **denied access** 🚨 error, do you think you can challenge yourself to:

* Find the **Permissions** tab in your new Lambda function
* Create a new **Resource-based policy statement** that gives your chatbot aliases access to your new function.
* Check whether your **CheckBalance** intent works after giving this a go!

Once you're done, don't forget to delete your CloudFormation stack!

* Head back to your **CloudFormation** console.
* Select the same stack that you created.
* In the Stack info pane, choose **Delete**.
* Choose **Delete** again. It takes a few minutes, but eventually the Stack list will return to 0 as you refresh your console.
* Still, check your **Lambda** console to delete any functions you've created manually.

Next, delete your CloudWatch log group. It wasn't created as a part of your CloudFormation stack, but it is a byproduct of testing your chatbot.

* Head to your **CloudWatch** console.
* Select the checkbox next to your **CloudWatch log groups.**
* In the **Actions** dropdown, select **Delete log group(s)**.